2022 年度 情報理工学実験 I

提出日:2022/10/23

工学部 情報エレクトロニクス学科 情報理工学コース 3年

氏名: 豊田真吾

学籍番号: 02206909

実験テーマ
~データベースとWebサーバプログラミング~
(Part1)

基本課題 2-1

Ruby で Fizz Buzz のプログラムを以下のように作成した。

FizzBuzz.rb

```
print "整数を入力してください:"
a = gets.to_i
#1からaまでfizzbuzz
puts "1から#{a}までのFizzBuzzを実行します"
for i in 1..a
    if i % 3 == 0 && i % 5 == 0
        puts "FizzBuzz"
    elsif i % 3 == 0
        puts "Fizz"
    elsif i % 5 == 0
        puts "Buzz"
    else
        puts i
    end
end
```

実行して結果を確認すると、以下のように標準入力から自然数を受け取って1からその自然数までのFizzBuzzの結果が表示される。

zsh

```
~/D/H/B/データベースとWebサービス構築 ❯❯❯ ruby FizzBuzz.rb
整数を入力してください:10
1から10までのFizzBuzzを実行します
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
```

基本課題 2-2

Prime, Socket, Webrick, CGI 以外 にもRubyには多数のライブラリが収録されており、'matrix'と呼ばれるライブラリでは行列に関するの計算を行うことができる。標準入力から正方行列の二次元配列を入力し、matrixを用いて行列式の値を出力するプログラムを以下のように作成した。

tryLibrary.rb

```
require 'matrix'
#正方行列を入力して2次元配列に保存
def input_matrix
```

```
    puts "行列のサイズを入力:"
    size = gets.to_i
    matrix = Array.new(size){Array.new(size)}
    puts "#{size}×#{size}の行列の要素を入力(数字ごとに改行):"
    size.times do |i|#行
        size.times do |j|#列
        matrix[i][j] = gets.to_i
        end
    end
    return matrix
end

m = input_matrix
#行列式の値を表示
puts "行列式の値は#{Matrix[*m].determinant}"
```

実行し、行列を入力すると以下のような結果が得られた。

zsh

```
~/D/H/B/データベースとWebサービス構築 ❯❯❯ ruby tryLibrary.rb
行列のサイズを入力:3
3×3の行列の要素を入力(数字ごとに改行):3
5
4
2
6
5
7
3
4
行列式の値は18
```

基本課題 2-3

次のように操作することでSocket 通信を行うプログラムを作成した。

1. minChat_Server.rb を実行する。
2. minChat_Client.rb を実行する。
3. minChat_Client.rb の標準入力で入力した任意の文字列が、minChat_Server.rb の出力に表示される。
4. minChat_Client.rb の標準入力で入力した文字列が"bye"であった場合には、"bye"がinChat_Server.rb の出力に表示された後双方のプロセスが終了する。

minChat_Server.rb

```
# coding: utf-8
require 'socket'
server = TCPServer.new 2000 #2000 番ポートにサーバを立てる
#クライアントの受付開始
client = server.accept #サーバのプログラムは，クライアントからのリクエストがあるまでここで一時停止(ブロッキングします)
while true
    msg = client.gets #クライアントから受信した文字列を表示
    puts msg
    if msg.chomp == 'bye' #受け取った文字列が bye ならば
```

```
        break #ループを抜ける
    end
end
client.close #クライアントとの通信を切断
```

minChat_Client.rb

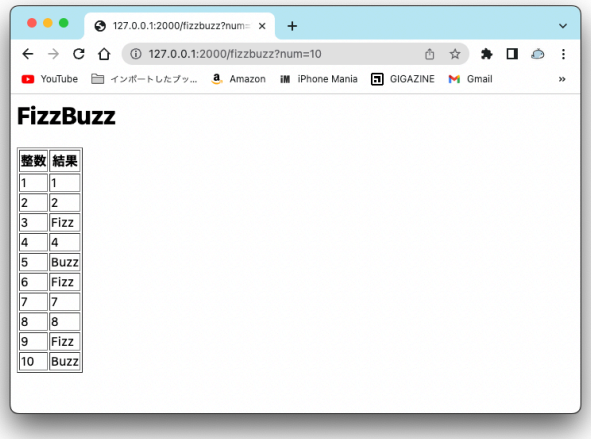
```
# coding: utf-8
require 'socket'
socket = TCPSocket.new('127.0.0.1', 2000) #127.0.0.1 の 2000 番ポートのサーバに接続

while true
    print 'Enter message : '
    msg = gets #標準入力から文字列を受け取る
    socket.puts msg #サーバに文字列を送信
    if msg.chomp == 'bye' #受け取った文字列が bye ならば
        break #ループを抜ける
    end
end
socket.close #サーバとの通信を切断
```

基本課題 2-4

Webrickを用いて現在時刻を表示するページを作成し、WebrickとCGIを用いてパラメータで与えられた値までのFizzBuzzを表示するページを作成した。127.0.0.1:2000/time にアクセスを行うと現在時刻を表示し、127.0.0.1:2000/fizzbuzz にアクセスすると、FizzBuzzの結果が表の形式で出力される。FizzBuzzについてはnumパラメータを与えることによって1から任意の自然数までのFizzBuzzの結果が表示されるようになっている。例えば、1から10までのFizzBuzzを出力したい場合は、127.0.0.1:2000/fizzbuzz?num=10
をブラウザで開くと良い。なおソースコードはここでは割愛し、別ファイルとして添付する。本パートの以降の課題についてもソースコードは割愛する。

127.0.0.1:2000/time



127.0.0.1:2000/fizzbuzz?num=10

基本課題 2-5

攻撃方法

XSS
入力された内容を表示する Web ページに攻撃者が攻撃用のスクリプトを仕組ませることで、別の Web サイトを表示させ、利用者の個人情報を盗む手法。
CSRF
攻撃者が Web サイトに罠を仕組ませ、それを利用者に閲覧させる。別の Web サイトで利用者を偽り、意図しない操作を行わせる攻撃手法。両社の違いは、XSSはWeb ブラウザ上で悪意のあるスクリプトを実行させ，CSRF はWeb サーバ上で悪意のあるスクリプトを実行させる点で違いがある。

攻撃例

XSS
攻撃者は正規の Web サイト上に用意したURLやスクリプトなどの、偽情報を提供する。
そのリンク先などで不正なスクリプトが実行されてしまうことで、利用者のCookieが攻撃者に送信されるなどの被害が発生する。

対策

XSS
管理者側はスクリプト上で無駄となる文字列を排除して運営時に定期的にセキュリティ監査を行う等の対策をすればよい。また、問題が発生したらウェブサイトの一部や全体の公開を一時停止する必要がある。SNSでは投稿できる内容に制限を加える必要がある。ユーザ側は、ウェブサイトにアクセスする場合は、信頼できる情報源から URL を取得する。2 回目に訪問する予定があればお気に入り登録をしておく。
CSRF
管理者側は悪意のある HTTP リクエストをWeb アプリケーションが受け付けないようにすればよい。ウェブサイトで行える操作によっては、問題の規模が異なるので規模が多きければ早急な対応が必要となる。ユーザ側はウェブサイトにログインして作業を行う場合は、必要な分だけの作業を行い、作業後はログアウトするようにすればよい。

XSS と CSRF 以外

SQLインジェクションについて
攻撃方法

アプリケーションの脆弱性により本来の意図ではない不当な SQL 文が作成されてしまい、注入されることによって、データベースのデータを不正に操作される攻撃のことである。
データベースと連携して動作する多くのWebアプリケーションは SQL 文を使い、ユーザからの各種情報入力を元にしてサーバ側で処理し結果をユーザに返す。
Webアプリケーションに脆弱性が存在すると、アプリケーションが入力値を適切に処理せず、SQL文を誤った命令文として認識してしまい、データベースの操作が可能になることで、情報漏えいやデータの悪用につながる。

攻撃例

例えば、WebサイトのID欄の入力値として 「myname」が入力されてSQLが実行されたとき、

```
SELECT * FROM user WHERE id='myname'
```

というSQL文が実行され情報を得ることができる。ここで、入力欄に「'or 'A'or A'」と入力することによって、実行されるSQL文は

```
SELECT * FROM user WHERE id='taro' or 'A'='A'
```

となり、全ユーザの情報を得ることができてしまう。

対策

プレースホルダを用いて実装を行うことでSQLインジェクションを回避することができる。具体的には、:から始まる代替文字列(プレースホルダ)を次のように宣言し、

```
$sql = "SELECT * FROM user WHERE name=:name";
```

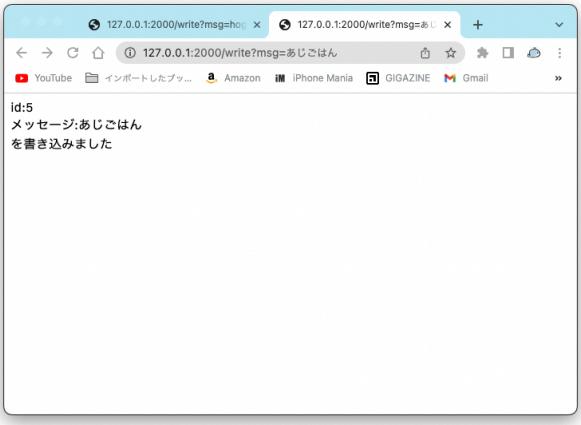
次のように後から実際の値をバインドさせればnameの部分をSQL文と解釈されなくなるため、SQLインジェクションを回避できる。

```
bindValue(':name', $name, PDO::PARAM_STR);
```

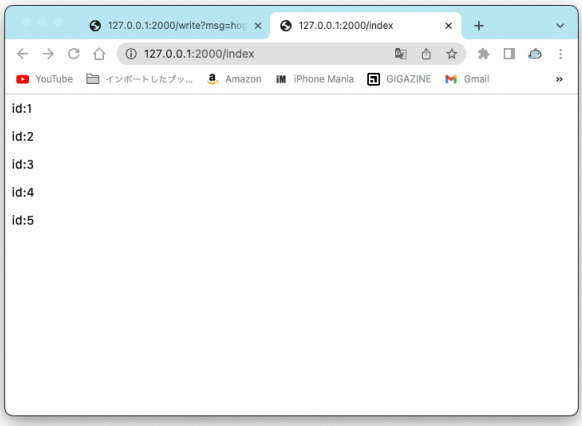
応用課題 2-1

次のような操作でテキストを保存・確認することができる掲示板プログラムを作成した。

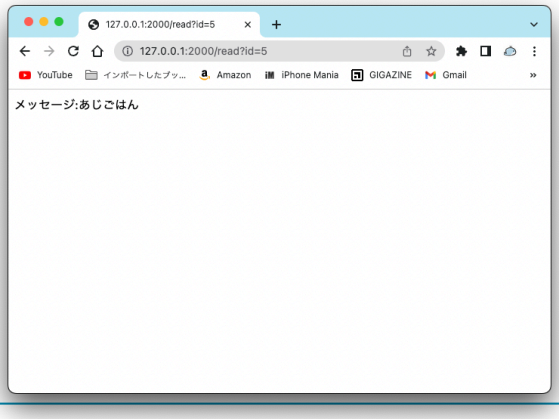
1. http://127.0.0.1:2000/write?msg=<任意の文字列> とすることで任意の文字列を書き込み、ユニークなIDを生成して表示



2. http://127.0.0.1:2000/index とすることで書き込みのid一覧を確認



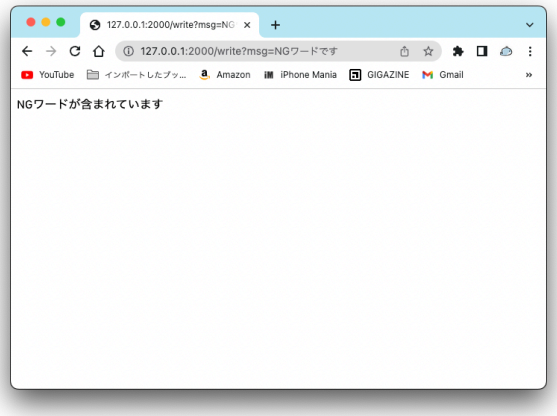
3. http://127.0.0.1:2000/read?id=<任意のID> とすることで任意のidのメッセージを表示



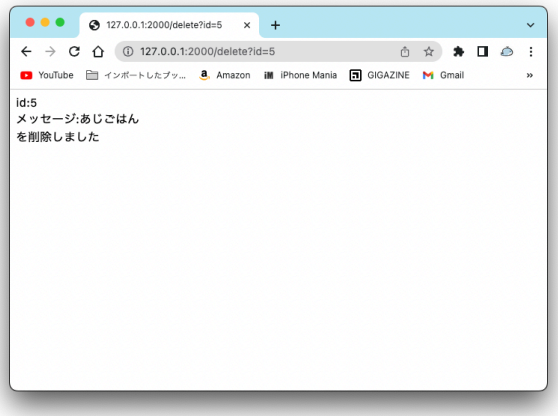
この他にも禁止ワード機能、削除機能を実装した。

4. writeで与えた文字列に禁止ワード配列に登録されている文字列が含まれていたら、NGワードが含まれている旨を表示し、保存を行わない。

```
NG = [",","\r","\n","NG"]
```



5. http://127.0.0.1:2000/delete?id=<任意のID> とすることでメッセージを削除する



データの保存方法について

data.txtというテキストファイルを用意して、

```
id,メッセージ
id,メッセージ
id,メッセージ
id,メッセージ
…
```

のようにカンマ区切りのidとメッセージの形式で保存している。

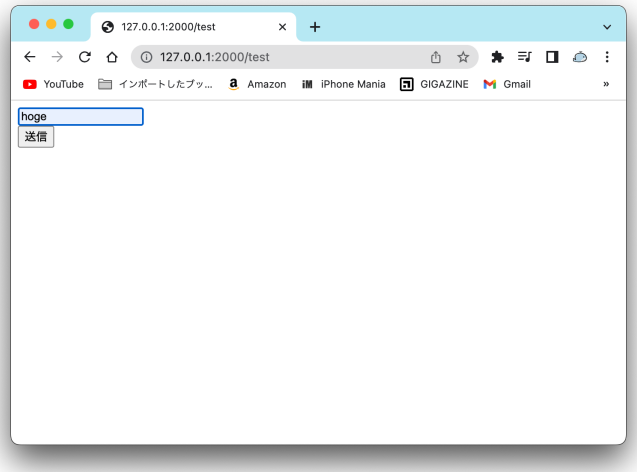
IDの決定方法について

data.txtの最終行のID+1を新しいメモのIDとすることで常にユニークなIDが割り当てられるようになっている。data.txtが空の場合は新規メッセージのIDは1となる。

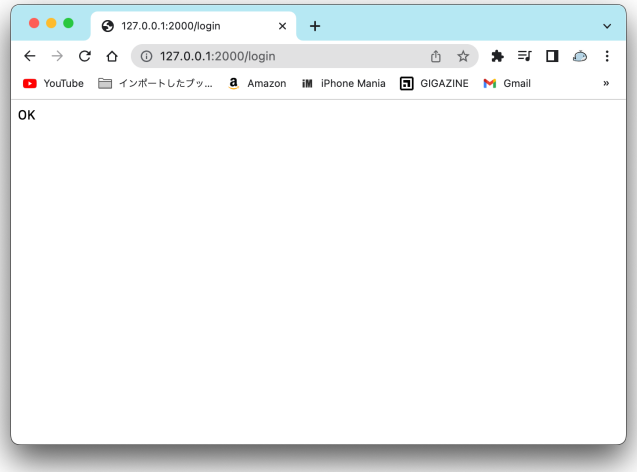
応用課題 2-2

次の手順でログインを行う。

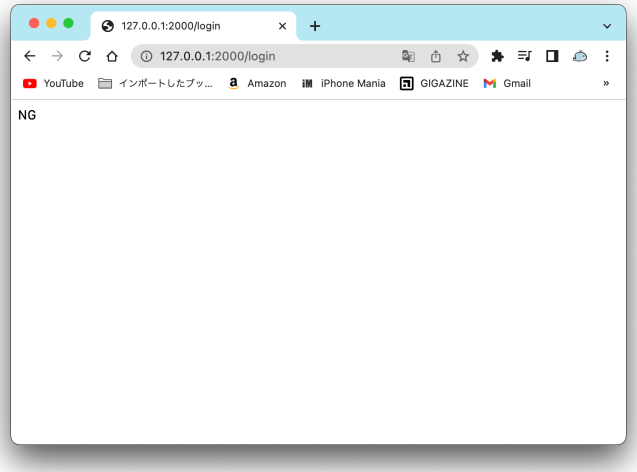
1. http://127.0.0.1:2000/test にアクセスする。
2. テキストフィールドに文字列を入力する。



3. 入力したテキストがlist.htmlのいずれかの行と一致していればOKと表示される。



4. 一致していなければNGと表示される。


127.0.0.1:2000/login
127.0.0.1:2000/login
YouTube
インポートしたブッ...
Amazon
iPhone Mania
GIGAZINE
Gmail
NG